Remarks

Claims 1 - 33 are pending in the application. Claims 1 - 21, 23 - 31 and 33 have been variously rejected under 35 U.S.C. §§ 102 and 103. Claims 1, 7, 13, 22, 27 and 32 have herein been amended and are fully supported by the specification. Applicant has amended the claims to clarify the claim language and reserves the right to further clarifications. No new matter has been added to the prosecution of this application. For at least the reasons stated below, Applicant asserts that all claims are now in condition for allowance.

1. Claims 22, 27 and 32 should now be Allowed

The Examiner held that claims 22, 27 and 32 would be allowable if rewritten in independent form. The three claims have been so amended. Applicants request that the three claims now be allowed.

2. 35 U.S.C. § 103 Rejections

All of the rejections are based in whole or in part on the Testardi patent (US Patent 6,249,882). Since Applicants assert that Testardi fails to teach the invention for several reasons, the Examiner's rejections for claim 19 will first be discussed. These arguments and remarks are then extended to the remaining rejected claims

To establish a *prima facie* case of obviousness "the prior art reference ... must teach or suggest **all** the Claim limitations." (emphasis added) (MPEP § 2143). Because the cited references alone or in combination fail to teach or suggest **all** the Claim limitations, Applicants respectfully request that the Examiner's §103 rejections be withdrawn.

With respect to claim 19, in section 3 of the Office Action the Examiner rejects the claim as being unpatentable over Testardi (US Patent 6,249,882) in view of Silva et al. (US Patent 6,014,760). As will be seen in the following discussion, there is no combination of teachings or suggestions between Tessardi and/or Silva that support a section 103 rejection of claim 19. Thus all rejections based on that independent claim, the related claims 24 and 29, and the related dependent claims 20-23, 25-28, and 30-33, should be withdrawn.

Claims 1 - 18 have been amended to clarify that the invention uses words having commonly understood meanings. Since the discussion of claim 19 shows that Tessardi and Silva neither teach nor suggest (alone or in combination) any such use of words having commonly understood meanings, the discussion of claim 19's rejection under section 103 also refutes the section 102 rejections for claims 1-18.

<u>Overview of Applicants' Invention</u>

After a software program is developed it must be tested to verify it meets certain requirements. Testing often proceeds through several phases. For example, software is sometimes subjected to module testing, string testing, system testing, regression testing, performance testing, and load testing.

Such testing is time consuming and repetitive. Regression testing, for example, requires a team of testers to use the system in mock conditions to verify that changes to the system did not change how the system handles certain primary activities. Load testing requires that a very large number of testers simultaneously use the system to see if it buckles under a load of use. Having a large team of testers can be impractical and/or cost prohibitive. As a solution to regression and load testing's problems of taking a long time and involving repetitive tasks, automated testing tools (such as the WINRUNNER and LOADRUNNER software packages) have been developed to emulate one or more users. After the testing tool is configured, it can robotically run the software – automatically entering text, moving the mouse, clicking buttons and choosing pull-down menus, all without actual human interaction.

Unfortunately, configuring automated testing tools is difficult since they rely on scripts that are programmed using a complicated scripting language. Applicants' invention is an interface situated <u>between</u> the tester and the automated testing tool. This interface allows the tester to build her own testing scripts without needing to understand the scripting language. Rather, the tester uses standard English terms having commonly understood meanings to instruct the interface in how the test run. The interface then converts these English terms into the proper scripting code. The interface can also accept words that are common understood in languages other than English. For example, the interface allows a Spanish speaking tester to use common Spanish terms to create test scripts.

As recited in claim 19, Applicants' invention uses commonly understood words to run the testing tool (to emulate a series of user interactions) by:

(1) <u>receiving</u> a word having a commonly understood meaning;

(2) <u>querying</u> a database for the word, the database containing a plurality of words in which each word has associated to it a set of computer instructions for a computer to perform the commonly understood meaning of the word;

(3) <u>retrieving</u> the instruction set corresponding to the word; and

(4) <u>using</u> the automated testing tool to perform the function that is related to the commonly understood meaning of the word.

In other words, claim 19 states that the tester enters an English-like sentence describing what the test should accomplish. Each common word in this sentence is translated by the invention into the proper grouping of script commands. The script commands are loaded into the testing tool. The testing tool then emulates a user by applying the proper inputs to the program being tested as if a real user were sitting at the computer keyboard entering the text, clicking the mouse, etc.

This type of testing – in which inputs are fed to the software and the results are inspected to see how the software responds – is known as black-box testing since the software's internal workings are invisible to the tester, as if the software is hidden in a <u>black</u> box.

<u>Overview of the Testardi Invention</u>

The Testardi invention focuses on the inclusion of <u>white</u> box testing in an automated testing environment. In white-box testing, the tester knows the inner workings of the software. She is usually one of the programmers that developed the software. Through her knowledge of the various modules, calls, branches, error checking, and the like, the white-box tester fashions tests specifically designed to make sure that these software structures work as planned. For example, the white-box tester may inspect a module of code and set up three tests – one to test when a necessary data file is properly configured, one to test when that data file is empty, and one to test when the data file contains corrupted data. By knowing how the software is designed to react to each of these three situations, the white-box tester ensures that it runs correctly.

<u>Testardi does not teach the "receiving" step of Claim 19</u>

Testardi (neither alone or in combination with Silva) does not teach or suggest each of the four components of claim 19. First, the Examiner asserts that Testardi teaches "**receiving a word having a commonly understood meaning**" in two sections – column 5 lines 39-65; and column 10 lines 4-15. This cannot be justified. The section found on column 5 discusses figure 1, which is a block diagram of the Testardi invention. According to column 5, that invention uses a program under test and an automated test tool on a computer. The automated test tool is operated to "extract test sequence information from **program source code** files" (lines 59-60). Then the test sequence extractor "**parses program source code files** corresponding to the program under test for purposes of **locating embedded test sequence descriptions** within the program source code files" (line 65 through column 6 line 2). In other words, column 5 teaches a testing tool that looks through source code files for special test sequence descriptions embedded in comment lines. This is not the same as "receiving a word having a commonly understood meaning."

To illustrate the difference, column 21 lines 60 – 65 shows the type of test descriptors that the Testardi invention extracts from source code. Test names are shown between a pair of "###" strings, such as "### GlobalTest ###." To verify or test the value of global data, the "==" test descriptor is used, such as "global == 10" to test whether that variable is set to 10. Column 22 lines 55 through 65 give an example of the text instructions embedded in a comment statement within some source code, namely:

```
/*
### globalmax() ###
global == 10
globalmax(1,2) == 10
globalmax(20,2) == 20
globalmax(1,30) == 30
*/
```

In this example, the only English word is "global". The Testardi invention does not react to the "global" because of its common meaning. Rather, since the word appears on the left side of "==", the Testardi invention verifies that the global variable named 'global' currently has the value of 10.

Moving on to the second section of Testardi that the Examiner asserts teaches "receiving a word having a commonly understood meaning" (i.e., column 10 lines 4 – 15), Applicant notes that it describes part of figure 4. That figure illustrates the invention's ability to perform white box testing. In the figure, element 400 parses "source code files corresponding to the program" and "extract[s] a desired test sequence." The user may request the test sequence by name. If the user requests a series of test sequences, then element 400 is repeated so it parses and extracts each test sequence in turn. This section again speaks to the embedded instructions, of the kind shown in column 22 lines 55 through 65:

```
/*
### globalmax() ###
global == 10
globalmax(1,2) == 10
globalmax(20,2) == 20
globalmax(1,30) == 30
*/
```

Such a series of scripting is not "receiving a word having a commonly understood meaning." Rather, it is finding a series of instructions that can be used by the testing tool to adequately white-box test the subsequent code.

<u>Testardi does not teach the "querying" step of Claim 19</u>

The Examiner holds that Testardi teaches querying a certain type of database at two locations, namely, column 11 line 5 to column 12 line 8, and column 21 line 40 to column 23 line 41. As claimed in the querying step, Applicants' invention queries a database for the word that has a commonly understood meaning. The query returns from the database a set of computer instructions. When the automation testing tool executes this set of instructions, the computer performs a function that is related to the commonly understood meaning of the word. For example, the words "click return button" could cause the testing tool to issue the necessary commands to emulate a user moving the mouse to the <return> button on the screen and then clicking the mouse to activate that button.

The first cited section (column 11 line 5 to column 12 line 8) by the Examiner makes no reference to any type of database, or to any type of converting a commonly understood word to instructions that cause a computer to perform the associated commonly understood action. Rather, this section discussed the use of the "xdb" debugger to execute the command script as part of white-box testing. One skilled in the art understands that the xdb debugger is one of a series of possible debuggers offered in the UNIX operating system. The Testardi explains – as one skilled in the art already understands – that such a debugger uses the script to "monitor and display the present value of global and local variable of interest to detect proper operation of the program in accordance with the test sequence" (column 10, lines 37 – 40). In particular, the "debugger tool is instructed to display selected variable values and the test executive is instructed to compare variable values and notify the user of any errors in the program operation from such comparisons (column 10, lines 40 – 44). Thus, in the example given by Testardi:

```
/*
### globalmax() ###
global == 10
globalmax(1,2) == 10
globalmax(20,2) == 20
globalmax(1,30) == 30
*/
```

The xdb debugger reacts to the this embedded script by testing to see whether the global variable named 'global' currently has the value of ten. If not, then the user is notified. Next, the xdb debugger is instructed to call the function named globalmax, using the parameter list of "1, 2". Upon the function's return, the debugger verifies that the return value has been set to 10. The globalmax function is called two more times and its return values are again tested each time.

As can be seen in this example, nothing in the embedded instructions has a commonly understood meaning -- but for the term 'global' which is being used as a variable name rather in its commonly understood meaning. The use of "###", calls to routines, parameter lists enclosed by parentheses, and "= =" to request a test all have no commonly understood meaning. And so none of these 'words' cause the computer to perform the task associated to a commonly understood meaning.

The second section (column 21 line 40 to column 23 line 41) cited by the Examiner shows information for white box and black box testing by the Testardi invention. The white box testing verbiage (discussed in column 21) has already been discussed and dismissed above.

Starting at the top of column 23, Testardi's black box testing is discussed. Applicable test directives for this type of testing have no commonly understood meaning either. For example: "###" is used to start a new test; "in:" and "in:<EOF>" is used to set up stdin for the next command; and "file: " is used to create a file with a specified name. By combining these test directives, Testardi can perform black box testing. An example of this type of testing is shown in column 23, namely:

```
/*
### id ###
subst: AMI whoami
exec: -un
out: @AMI
out: <eof>
err: <eof>
fail:  getuid 0
exec: -un
out: root
out: <eof>
err: <eof>
*/
```

Nothing in the above example is a word having acommonly understood meaning. Thus, Applicant submits that neither section of Testardi teaches or suggests the "querying" step of claim 19.

<u>Testardi does not teach the "retrieving" step of Claim 19</u>

The Examiner holds that Testardi teaches "retrieving the instruction set corresponding to the (commonly understood) word from the database" at column 10 lines 4 - 31. Applicant's reading of column 10 shows that it discussed extracting the test sequences from comment statements within source code programs. Such test sequences have been discussed above and are not commonly understood words, but are rather such arbitrary symbols like "###" and "==". Column 10's description says that the symbols like "==" are then used to construct the command script provided to the xdb debugger for such purposes as setting variables, invoking procedures and displaying variable values. As is well known in the art, the xdb debugger uses a terse vocabulary. For example, the command "sr" instructs the debugger to display special registers, "lm" lists all string macros, "U" refreshes the windows, and "?[*string*]" searches backward for the term *string*. These are not commonly understood words.

As the test directives embedding in the comment statements – such as "###" and "==" – are not commonly understood words, and as column 10 discusses using such symbols to cause the xdb to perform tasks, and as xdb uses an arcane command language that does not consist of commonly understood words, Testardi does not teach "retrieving the instruction set corresponding to the (commonly understood) word from the database."

<u>Testardi does not teach the "using" step of Claim 19</u>

The Examiner holds that column 10 lines 32 – 44 of Testardi teaches "performing the function that is related to the commonly understood meaning of the word using the automated testing tool." The cited portion of column 10 discusses invoking the xdb debugger and providing it with a command script. The debugger then executes the commands and thereby "monitors and displays the present value of global and local variables of interest." As discussed above, the xdb debugger executes commands such as ""sr" to display special registers, "lm" to list all string macros, "U" to refresh the windows, and "?[*string*]" to search backward for the term *string*. Clearly, none of these activities are "related to the commonly understood meaning of the word" since terms like "sr" "lm" "U" and "?[ ]" have no such common meaning.

<u>Silva does not teach the database of Claim 19</u>

The Examiner admits that Testardi does not teach the use of a database, but that Silva does teach a database and combining the two references suggests or teaches the Applicants' invention. While it is true that the Silva invention includes a database library, Testardi fails to teach any of the four components of claim 19. There is no combination of Silva and Testardi together that teaches or suggests each and every element of claim 19, as is required for a rejection under section 103.

Since these references do not suggest or teach receiving, querying, retrieving, or performing, Applicants submit that the 103 rejection is improper for claim 19 and asks that it be withdrawn.

3. The § 102 rejections for claims 1 – 18 should be withdrawn

Claims 1, 3, 5-7, 9, 11-13, 15, 17 and 18 have been rejected under 35 U.S.C. § 102 as being anticipated by Testardi. For a proper § 102 rejection, a "claim is anticipated only if each and every element as set forth in the claim is found, either expressly or inherently described, in a single prior art reference." Verdegaal Bros. v. Union Oil Co. of California, 814 F.2d 628, 631, 2 USPQ2d 1051, 1053 (Fed. Cir. 1987).

Claims 1, 3, 5-7, 9, 11-13, 15, 17 and 18, as amended, are analogous to claim 19 to the extent that they feature the use of "words having commonly understood meanings". As the above discussion has shown, Tessardi neither teaches nor suggests such a use of commonly understood words and therefore fails to show each and every element of these claims. Thus, Applicants submit that the Examiner's § 102 rejections are unsupported by the art and should be withdrawn.

4. The § 103 rejections for claims 19 – 33 should be withdrawn

Claims 2, 4, 8, 10, 14, 16, 20, 21, 23 – 25, 26, 28 – 30, 31 and 33 all depend on claim 19 or one of the other independent claims that require the use of words having a commonly understood meaning as part of an interface with an automated testing tool. As it has been shown by the above discussion that Tessardi is an improper reference for Applicants' invention that utilizes words having commonly understood meanings, all of the these section 103 rejections are improper and should be withdrawn.

5. Conclusion

Applicants submit that all pending claims are allowable and respectfully requests that a Notice of Allowance be issued in this case. In the event a telephone conversation would expedite the prosecution of this application, the Examiner may reach the undersigned at 612-607-7508. If any fees are due in connection with the filing of this paper, the

Commissioner is authorized to charge such fees including fees for any extension of time, to Deposit Account No. 50-1901 (Reference 60021-355001).

Respectfully submitted,

[illegible]

Steven C. Lieske, Reg. No. 47,749
Customer No. 29,838

OPPENHEIMER WOLFF & DONNELLY LLP
1400 Page Mill Road
Palo Alto, CA 94304
Phone: 612-607-7508
Fax: 612-607-7100
E-mail: SLieske@oppenheimer.com